package com.twitter.search.earlybird\_root;

import java.util.concurrent.Executors;

import java.util.concurrent.ScheduledExecutorService;

import javax.annotation.Nullable;

import javax.inject.Named;

import javax.inject.Singleton;

import com.google.common.annotations.VisibleForTesting;

import com.google.common.util.concurrent.ThreadFactoryBuilder;

import com.google.common.util.concurrent.TwitterRateLimiterProxyFactory;

import com.google.inject.Provides;

import com.twitter.app.Flag;

import com.twitter.app.Flaggable;

import com.twitter.common.util.Clock;

import com.twitter.inject.TwitterModule;

import com.twitter.search.common.decider.SearchDecider;

import com.twitter.search.earlybird\_root.filters.ClientIdArchiveAccessFilter;

import com.twitter.search.earlybird\_root.filters.ClientIdQuotaFilter;

import com.twitter.search.earlybird\_root.filters.DisableClientByTierFilter;

import com.twitter.search.earlybird\_root.quota.ConfigBasedQuotaConfig;

import com.twitter.search.earlybird\_root.quota.ConfigRepoBasedQuotaManager;

public class QuotaModule extends TwitterModule {

@VisibleForTesting

public static final String NAMED\_QUOTA\_CONFIG\_PATH = "quotaConfigPath";

public static final String NAMED\_CLIENT\_QUOTA\_KEY = "clientQuotaKey";

private static final String NAMED\_REQUIRE\_QUOTA\_CONFIG\_FOR\_CLIENTS

= "requireQuotaConfigForClients";

private final Flag<String> quotaConfigPathFlag = createMandatoryFlag(

"quota\_config\_path",

"",

"Path to the quota config file",

Flaggable.ofString());

private final Flag<String> clientQuotaKeyFlag = createFlag(

"client\_quota\_key",

"quota",

"The key that will be used to extract client quotas",

Flaggable.ofString());

private final Flag<Boolean> requireQuotaConfigForClientsFlag = createFlag(

"require\_quota\_config\_for\_clients",

true,

"If true, require a quota value under <client\_quota\_key> for each client in the config",

Flaggable.ofJavaBoolean());

@Provides

@Singleton

@Named(NAMED\_QUOTA\_CONFIG\_PATH)

String provideQuotaConfigPath() {

return quotaConfigPathFlag.apply();

}

@Provides

@Singleton

@Named(NAMED\_CLIENT\_QUOTA\_KEY)

String provideClientQuotaKey() {

return clientQuotaKeyFlag.apply();

}

@Provides

@Singleton

@Named(NAMED\_REQUIRE\_QUOTA\_CONFIG\_FOR\_CLIENTS)

boolean provideRequireQuotaConfigForClients() {

return requireQuotaConfigForClientsFlag.apply();

}

@Provides

@Singleton

ClientIdQuotaFilter provideConfigRepoBasedClientIdQuotaFilter(

ConfigRepoBasedQuotaManager configRepoBasedQuotaManager,

TwitterRateLimiterProxyFactory rateLimiterProxyFactory) throws Exception {

return new ClientIdQuotaFilter(configRepoBasedQuotaManager, rateLimiterProxyFactory);

}

@Provides

@Singleton

ConfigBasedQuotaConfig providesConfigBasedQuotaConfig(

@Nullable @Named(NAMED\_QUOTA\_CONFIG\_PATH) String quotaConfigPath,

@Nullable @Named(NAMED\_CLIENT\_QUOTA\_KEY) String clientQuotaKey,

@Nullable @Named(NAMED\_REQUIRE\_QUOTA\_CONFIG\_FOR\_CLIENTS) boolean requireQuotaConfigForClients,

Clock clock

) throws Exception {

ScheduledExecutorService executorService = Executors.newSingleThreadScheduledExecutor(

new ThreadFactoryBuilder()

.setNameFormat("quota-config-reloader")

.setDaemon(true)

.build());

return ConfigBasedQuotaConfig.newConfigBasedQuotaConfig(

quotaConfigPath, clientQuotaKey, requireQuotaConfigForClients, executorService, clock);

}

@Provides

@Singleton

DisableClientByTierFilter provideDisableClientByTierFilter(

ConfigRepoBasedQuotaManager configRepoBasedQuotaManager,

SearchDecider searchDecider) {

return new DisableClientByTierFilter(configRepoBasedQuotaManager, searchDecider);

}

@Provides

@Singleton

ClientIdArchiveAccessFilter clientIdArchiveAccessFilter(

ConfigRepoBasedQuotaManager configRepoBasedQuotaManager) {

return new ClientIdArchiveAccessFilter(configRepoBasedQuotaManager);

}

}